HMEE325 – ROS
PROJECT REPORT

AUTHOR: EGE GURSOY

PROFESSOR: BENJAMIN NAVARRO

Objective

In this project we present the turtlesim simulator. Our goal is to draw a snake shaped line using the turtle.

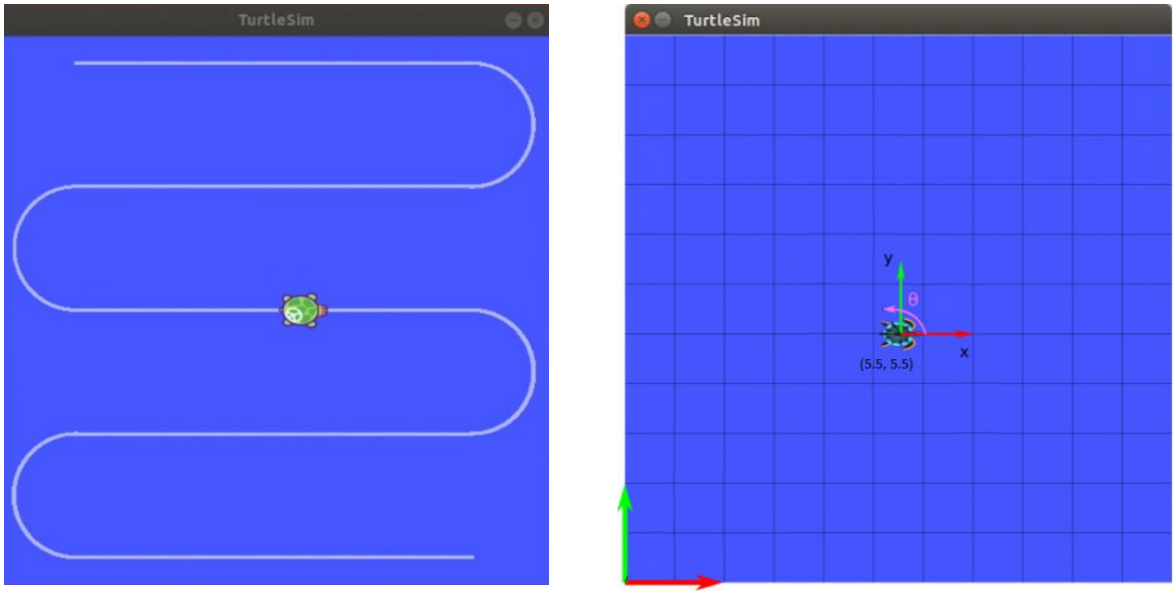


Figure : The snake shape (left) and the turtlesim coordinate system (right)

Turtlesim

Turtlesim is a built-in package of the ROS. It is a turtle in a 11 x 11 board which can be controlled by using existing topics. The coordinate frame of the turtlesim is described below. At the initial launch, turtle spawn at the middle of the window (5.5, 5.5).

Node

To automatically draw a shape with the turtle, we created a node. The node follows the FSM architecture. It has the required components to subscribe, to publish and to switch between its states.

Communication

To draw the shape, we need to subscribe to the position and publish on the velocity of the turtle. Turtlesim has the “turtlesim::Pose” message for the positions and the “geometry_msgs::Twist” message for the velocities. The Pose and Twist belong to the “turtle1/pose” and “turtle1/cmd_vel” topics, respectively.

We created a callback function with the Pose. Then used that function to create a subscriber to the “turtle1/pose”. To command the turtle, we wrote a publisher to “turtle1/cmd_vel” which uses Twist messages.

We can access the positions by typing position.x, position.y, position.z. The “position” variable here is a “turtlesim::Pose” type variable.

The “geometry_msgs::Twist” types has an angular and linear component. For each component, we have x,y,z components at our disposal.

It is important to note that we can only command the linear velocity on the x axis and the angular velocity on the z axis.

States

Since we are using a FSM architecture, we have states to express the behavior of the turtle. The turtle begins at the Forward state. We defined some threshold values for the turtle to be considered at the top, bottom, left or right of the window.

Forward

In this state, turtle moves forward with a linear velocity on the x axis. To do that, we publish constantly a linear velocity on the x axis.

Once the x position exceeds a the left or right threshold, the turtle considered at the left or at the right. In that case, turtle passes to the “Turn Positive / Negative” states.

If however, the turtle exceeds the top and the left thresholds, or the bottom and the right thresholds at the same time, the turtle proceeds to the “Turn fixed” state.

If the turtle aims the bottom right part of the shape, it passes to the “Turn negative” state when it arrives at the right and to the “Turn positive state” when it arrives at the left. If it aims the top left part of the shape, it does the vice-versa. We used a boolean to represent the top left or the bottom right direction. From now on we will refer the bottom right direction as BR and top left direction as TL.

Turn positive/negative

Positive and negative words used to represent the turning direction. We assume the anti-clockwise direction as positive. When the turtle arrives at the left or right side of the window, it should turn. If it goes towards the BR direction, it should take a negative turn at the right side and a positive turn at the left side and vice-versa for the TL direction.

To do that, once the turtle arrives at the right or the left side, we publish a twist message to send a linear velocity on the x axis and an angular velocity on the z axis.

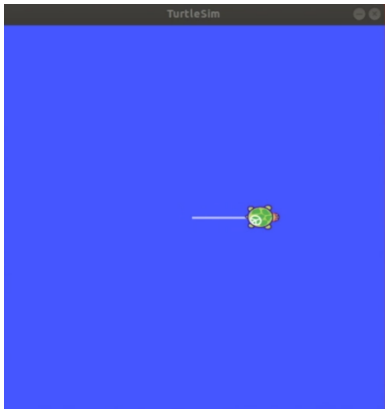
After publishing the messages, the turtle goes to the next state, which is the angle decision.

Turn fixed

If the turtle arrives at the bottom left or the top left corner of the window, then it should perform a 180 degrees turn to change its direction. We have to check this situation before the turn positive/negative state. Otherwise, the turtle would never enter to the turn fixed state.

In the turn fined case, we set the linear velocity to zero and angular velocity on the z axis to desired turning speed then publish the twist message.

After publishing the messages, the turtle goes to the next state, which is the angle decision.



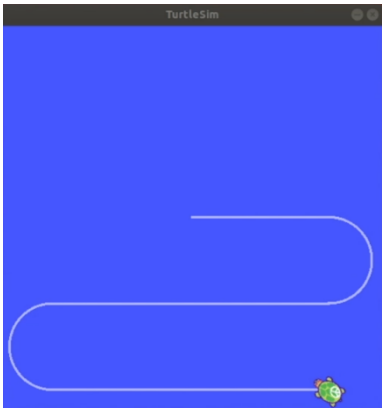


Figure : Forward (left) and turn fixed (right) states

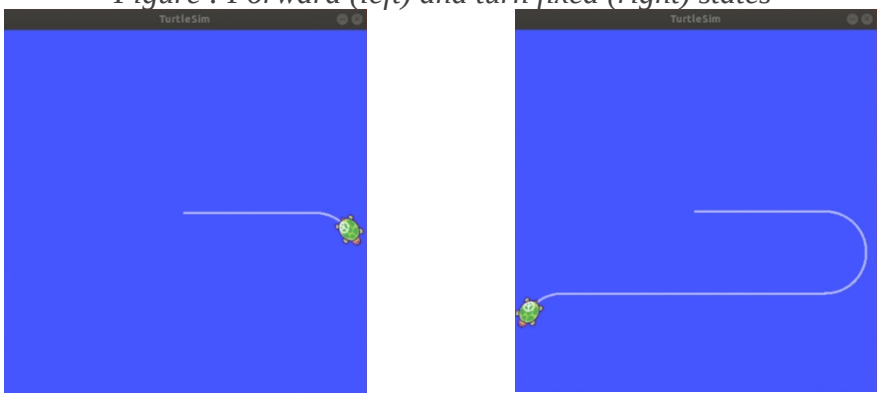


Figure : Turn negative (left) and turn positive (right) states

Angle decision

At this state, we decide to the angle which the turtle stops turning. In all the turnings, the turtle performs a 180 degrees turn, either starting from the 0 degrees or 180 degrees of head angle. If the turtle starts the turn at the 0 degrees, it should stop at 180 degrees (or pi radian) and vice-versa.

Since our command does not verify if the turtle is exactly at 0 degrees or 180 degrees, we can only assume that it is around these values. To find out if the turtle is around 0 degrees or 180 degrees, we set a threshold (or delta) value. If the head angle that we get from the pose message is under that threshold, we assume the turtle is at 0 degrees and if it is between 180 + threshold and 180 – threshold, we assume the turtle is at 180 degrees.

According to the found angle is 0, the turtle goes to the “To pi” state and if it is 180, it goes to the “To zero” state.

To pi/zero

Pi and zero are the angles in radian that we want to stop the turtle. We constantly read the head angle information from the pose message and when the turtle is between the angle – threshold and angle + threshold, we assume the angle as pi or zero depending on the state. It is crucial to increase the loop rate of the node in order to read the pose messages more frequently to stop at the right moment.

After reaching the right angle, the turtle returns to the forward state.